IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	JARABEK, Andrew, Louis; TOMBUL, Aris; LAU, Warren, H.
Serial No.:	10/675,097
Filed:	September 30, 2003
Title:	MEMORY INTEGRITY SELF CHECKING IN VT/TU CROSS-CONNECT
Group:	2138
Examiner:	RADOSEVICH, Steven D
Attorney Ref.:	PAT 2509-2 US

December 12, 2006

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Applicant submits the following Appeal Brief pursuant to 37 C.F.R. § 41.37 for consideration by the Board of Patent Appeals and Interferences. **The Commissioner is hereby authorized to debit $500.00 from Deposit Account No. 501593, in the name of Borden Ladner Gervais LLP, representing the fees for filing the opening brief as required by 37 C.F.R. §41.20(b).** The Commissioner is hereby authorized to charge any additional fees, and credit any over payments to Deposit Account No. 501593, in the name of Borden Ladner Gervais LLP. A duplicate copy of the Fee Transmittal is enclosed for this purpose.

TABLE OF CONTENTS

I. REAL PARTY IN INTEREST

The real party in interest is the assignee, Nortel Networks Limited.

II. RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences known to the appellants, the appellants' legal representative, or assignee, which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

III. STATUS OF CLAIMS

Claims 9, 10, 19, 20 and 27, 28 have been cancelled from the application. Claims 1-8, 11-18 and 21-26 are pending and presently stand rejected.

IV. STATUS OF AMENDMENTS

No Claim Amendments were made after the final rejection. Arguments were submitted in response to the Final Office Action.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The claims of the present application all relate to a scheme for detecting errors in a memory. More specifically, the claims are directed to a method for detecting the corruption of data stored in memory by comparing a first checkword generated before the data is written into the memory, to a second checkword generated after the data is read out of the memory. The first and the second checkwords are stored in a write accumulator and a read accumulator, respectively. If the first and the second checkwords are the same, then the data read from the memory is determined to be the same as the data originally written to the memory. On the other hand, if the first and the second checkwords are different, then the data read from the memory is determined to be corrupted.

A summary of the invention recited in independent claims 1, 11, 13 and 21 will now follow with reference to the description and drawings of the present application, as published in US2005/0071718A1.

Claim 1

Independent claim 1 is directed to a method for detecting errors. The method of claim 1 is

illustrated by an embodiment of the invention shown in the flow chart of Figure 3, which is described at paragraph [0018] of the description. The description of paragraph [0018] refers to elements of the switching core 200 shown in Figure 2, which is itself described at paragraph [0017]. Method claim 1 is reproduced for ease of reference, and includes bracketed reference numbers corresponding to the steps outlined in the embodiment of Figure 3 and the referenced switching core elements of the embodiment shown in Figure 2:

1. A method including:

(64) generating a first check word based on incoming data;

(66) storing the first check word in a write accumulator (272);

(70) generating a second check word based on stored data;

(72) storing the second check word in a read accumulator (273);

(74) comparing the first check word to the second check word by way of a compare circuit (274) connected to the write accumulator (272) and the read accumulator (273);

(76) generating a comparison result; and

indicating a failure based on the comparison result.

Paragraph [0018] details the switching core embodiment for the method recited in claim 1:

> *"the data input line 201 is tapped by a data line 205 and process 60 reads the data from the data in line 205. Process 60 generates (64) a first check word based on specific bytes of data lines 205 and stores (66) the check word in a write accumulator 272. Subsequent to storing the data in the RAMs 241, 242, 243, 244, and 245, process 60 reads (68) selected memories and locations of the memory associated with the bytes of data used to generate the first check word. … Process 60 generates (70) a second check word based on the data read from the memories and stores (72) the second check word in a read accumulator 273. Process 60 compares (74) the contents of the write accumulator 46 and the read accumulator 273 using compare circuit 274. Process 60 outputs (76) a signal based on the comparison. If the check words do not match, the signal alarms the host processor indicating a failure via output line 204."*

Claim 11

Independent claim 11 is directed to a method for detecting errors, similar to the method recited in claim 1. The method of claim 11 is illustrated by an embodiment of the invention shown in the flow chart of Figure 3, which is described at paragraph [0018] of the description. The description of paragraph [0018] refers to elements of the switching core 200 shown in Figure 2, which is itself described at paragraph [0017]. Method claim 11 is reproduced for ease of reference, and includes bracketed reference numbers corresponding to the steps outlined in the embodiment of Figure 3 and the referenced switching core elements of the embodiment

shown in Figure 2:

11. A method comprising:

(64) generating a first check word based on incoming data to a subset of a plurality of memories (241, 242, 243, 244 and 245);

(66) storing the first check word in a write accumulator (272);

(68) reading a set of data stored in the subset of the memories (241, 242, 243, 244 and 245);

(70) generating a second check word based on the set of data;

(72) storing the second check word in a read accumulator (273);

(74) comparing the first check word to the second check word by way of a compare circuit (274) connected to the write accumulator (272) and the read accumulator (273);

(76) generating a comparison result; and

indicating a failure based on the comparison result.

One main difference between claims 1 and 11 is that claim 11 recites the additional step of "(68) reading a set of data stored in the subset of the memories". Paragraph [0018] details the switching core embodiment for the method recited in claim 11:

> *"the data input line 201 is tapped by a data line 205 and process 60 reads the data from the data in line 205. Process 60 generates (64) a first check word based on specific bytes of data lines 205 and stores (66) the check word in a write accumulator 272. Subsequent to storing the data in the RAMs 241, 242, 243, 244, and 245, process 60 reads (68) selected memories and locations of the memory associated with the bytes of data used to generate the first check word. … Process 60 generates (70) a second check word based on the data read from the memories and stores (72) the second check word in a read accumulator 273. Process 60 compares (74) the contents of the write accumulator 46 and the read accumulator 273 using compare circuit 274. Process 60 outputs (76) a signal based on the comparison. If the check words do not match, the signal alarms the host processor indicating a failure via output line 204."*

Claim 13

Independent claim 13 is directed to a computer program on a computer readable medium with instructions for checking memory contents. The instructions of claim 13 correspond to the method steps recited in claim 1. The recited instruction steps are illustrated by an embodiment of the invention shown in the flow chart of Figure 3, which is described at paragraph [0018] of the description. The description of paragraph [0018] refers to elements of the switching core 200 shown in Figure 2, which is itself described at paragraph [0017]. Method claim 13 is reproduced for ease of reference, and includes bracketed reference numbers corresponding

to the steps outlined in the embodiment of Figure 3 and the referenced switching core elements of the embodiment shown in Figure 2:

13. A computer program product tangibly embodied on a computer readable medium, for checking contents of a memory in network switching environment comprising instructions for causing a computer to:

(64) generate a first check word based on incoming data;

(66) store the first check word in a write accumulator (272);

(70) generate a second check word based on stored data;

(72) store the second check word in a read accumulator (273);

(74) compare the first check word to the second check word by way of a compare circuit (274) connected to the write accumulator (272) and the read accumulator (273);

(76) generate a comparison result; and

indicate a failure based on the comparison result.

Reference can be made to previously excerpted paragraph [0018], which illustrates the features steps of claim 13.

Claim 21

Independent claim 21 is directed to a computer program on a computer readable medium with instructions for checking contents of a set of memories. The instructions of claim 21 substantially correspond to the method steps recited in claim 13. The recited instruction steps are illustrated by an embodiment of the invention shown in the flow chart of Figure 3, which is described at paragraph [0018] of the description. The description of paragraph [0018] refers to elements of the switching core 200 shown in Figure 2, which is itself described at paragraph [0017]. Method claim 21 is reproduced for ease of reference, and includes bracketed reference numbers corresponding to the steps of the embodiment outlined in Figure 3 and the referenced switching core elements of the embodiment shown in Figure 2:

21. A computer program product tangibly embodied on a computer readable medium, for checking contents of a set of memories in network switching environment comprising instructions for causing a computer to:

(62) store data in multiple memories (241, 242, 243, 244 and 245);

(64) generate a first check word based on incoming data to a predetermined subset of the memories (241, 242, 243, 244 and 245);

(66) store the first check word in a write accumulator (272);

(68) read a set of data stored in the predetermined subset of the memories (241, 242, 243, 244 and 245);

(70) generate a second check word based on the set of data;

(72) store the second check word in a read accumulator (273);

(74) compare the first check word to the second check word by way of a compare circuit (274) connected to the write accumulator (272) and the read accumulator (273);

(76) generate a comparison result; and

indicate a failure based on the comparison result.

Reference can be made to previously excerpted paragraph [0018], which illustrates the features steps of claim 13.

VI. ISSUE

Whether claims 1 and 13 are unpatentable under 35 USC 103 over U.S. Patent No. 4,171,765 (referred to hereafter as Lemone) in view of U.S. Patent No. 6,928,607 (referred to hereafter as Loaiza), and whether claims 11 and 21 are unpatentable under 35 USC 103 over Lemone in view of Loaiza and further in view of U.S. Patent No. 6,388,920 (referred to hereafter as Katayama).

VII. ARGUMENTS

The Appellants contend that claims 1, 11, 13 and 21 are unobvious and patentable over Lemone in view of Loaiza, and over Lemone in view of Loaiza and Katayama, for two primary reasons. First (i), Appellants submit that there is insufficient motivation for a person skilled in the art to combine the teachings of Lemone and Loaiza. Second (ii), Appellants submit that the combination of Lemone and Loaiza fail to disclose all the recited features of the independent claims. As both these two criteria must be required for establishing a *prima facie* case of obviousness as outlined in MPEP 706.02(j), Appellants submit that the obviousness rejection under 35 USC 103 in view of Lemone and Loaiza is improper and should be withdrawn. Since the Examiner's rejection to claims 1, 11, 13 and 21 relies on a base

combination of Lemone and Loaiza, the Examiner's rejection in view of Lemone, Loaiza and Katayama is inherently improper.

(i) Insufficient motivation

In the Advisory Action dated September 27, 2006, the Examiner reasoned that Loaiza suggests motivation for combining his teachings with Lemone:

> *"The motivation to combine was clearly present in the Loaiza U.S. Patent 6928607 B2 and Lemone U.S. Patent 4141765 references. One such motivation would be if the Loaiza 6928607 B2 patent "discloses or suggests in any manner [,the] reducing of overhead within storing and retrieving data from a disk." Such is found at as is described in detail above and in column 3, lines 7-9. Loaiza states, "…there is a need for … reducing the overhead that is typically associated with storing and retrieiving data …" We find that as described above Loaiza achieves the reduction in overhead while maintaining fundamental principles of the invention is a specific motivation."*

The Examiner later states that further motiviation lies in the similar nature of the problem to be solved by Lemone and Loaiza. In particular, the Examiner understands that Lemone desires to simplify and reduce the error detection hardware, and that Loaiza desires to reduce overhead associated with storing and retrieiving data.

Appellants submit that the motivation suggested by Loaiza is applicable only to reducing overhead related to performing logical checks in the data integrity verification system, and is not related to performing physical checksums. The teachings of Lemone and the presently claimed invention are directed to verification systems which only use physical checksum schemes. The logical check and the physical checksum schemes are performed differently and with different circuit components. Therefore, the reduced logical check overhead scheme of Loaiza is not applicable in any way to Lemone or the presently claimed invention. Therefore, a skilled person in the art could not be motivated to combine the reduced overhead logical check teachings of Loaiza with the physical checksum based system of Lemone, as Lemone does not use logical checks for validating data integrity. A description of the Loaiza and Lemone data verification schemes follows.

Loaiza is directed to a system for validating the integrity of data written to a non-volatile storage means, such as a disk drive. Loaiza teaches that two distinct schemes must be used for identifying corrupted data that has been stored to the disk drive. One is a physical

checksum and the other is a logical check. An example of a physical checksum operation is discussed by Loaiza at column 2, lines 33-38:

> *"For example, a checksum value may be calculated and stored within the data block so that when a logical operation, such as an exclusive-or (XOR) operation, is applied to the bits within the data block, a checksum constant such as zero is calculated."*

Use of the checksum constant is described later at column 2, lines 41-49:

> *"Afterwards, if application 104 again requires the updated information contained in the data block, the data block must again travel through several layers (i.e., from the one or disks 114-118 to disk array 110 and from disk array 110 to disk controller 106 over network 108) before it can be used by application 104. To determine the integrity of the data block, a physical checksum verification process is performed to verify that the data block still has the correct checksum constant value."* (underlining added)

Accordingly, some dedicated circuitry, such as XOR logic circuitry, would be used for generating the checksums. The logical check on the other hand, differs substantially from the physical checksum scheme, as it does not involve the generation of a code or additional information based on the data itself. The logical check is defined by Loaiza at column 2, lines 9-20:

> *"A logical check is a mechanism whereby the integrity of the data is determined by comparing the data to certain predetermined characteristics that are expected to be associated with the data values. For example, if a column in table A includes a set of pointers that are to index a specific row of table B, if any of the pointers has an address value that is not associated with a row of table B, that pointer may be identified as having a corrupted address value. Similarly, if a particular column in a table is configured for storing employee telephone numbers, if the value in any row for that column is determined to be negative, that value can be identified as corrupted."*

Accordingly, it is understood that the logical check involves a comparison of the data being written, against preset data. Loaiza opines the fact that the prior art scheme of performing the logical check twice requires considerable overhead, as stated at column 2, lines 56-64:

> *"However, a drawback with the described method for verifying the integrity of a data block is that performing a logical check on the information within the data block requires a significant amount of time and resources (overhead). For many applications that require a large number of data blocks to be continually written and read from disk, for example database applications, the additional overhead can dramatically affect the efficiency and response time of the application."*

Later at column 6, lines 10-14, Loaiza states an advantage of his novel scheme of Figure 2 for addressing the overhead problem of the prior art:

> *"Advantageously, because the physical checksum calculation 220 was performed prior to logical check 222, an additional logical check is not required to verify the*

> *integrity of the data block. Thus, the data retrieval overhead that is typically required in conventional systems can be significantly reduced."*

Therefore, it should be clear to any person skilled in the art that the "overhead" reduction taught by Loaiza is specific to the logic check. More specifically, it is not the writing to or retrieval of the data from the storage device that introduces overhead. It is the logic check executed upon the data which can delay transmission of the data to the application requesting the data. It is noted that the prior art physical checksum scheme used by Loaiza is still used in the same way in his improved scheme. Furthermore, it is noted that the physical checksum scheme of Loaiza is similar to the "error word" generation scheme used by Lemone.

Lemone is directed to a system for detecting and correcting errors using an error correcting code (ECC). Generally, the error detecting and correcting system of Lemone provides an error word based on data to be written to a storage device, in accordance with a polynomial function. The error word is then written with the data onto a storage device. A "read" error word is generated upon reading of the same data from the storage device, and is compared to the stored error word. Hence, if the "read" error word and the stored error word match, then the data is considered to be valid. The Examiner has summarized these teachings of Lemone in the Final Office Action. Lemone addresses the problem that complex error detection circuitry is required for implementing error detection and correction schemes. Lemone states the problem to be solved at column 1 lines 51-60:

> *"Generally, such error detection circuitry requires relatively complex multiplexing logic and further complex polynomial selection and generation logic requiring a relatively large number of logical elements. Such complexity not only increases the cost of the error detection circuitry but also decreases its reliability. It is desirable, therefore, to simplify as much as possible the error detection circuitry configuration so that hardware complexity and cost considerations can be considerably reduced."*

Lemone proposes to solve this problem by employing a programmable logic array, as summarized at column 1, line 63 to column 2, lines 1-7.:

> *"In accordance with the invention, an error detection system for use in detecting errors in serial data which has been read out from a data storage device is greatly simplified by performing the complicated multiplexing and polynomial generation operations with the use of a programmed logic array, the inputs to which are selected to produce desired selected outputs in a manner which greatly simplifies the error detection hardware and reduces the cost of the overall error detection system. The use of a programmed logic array not only eliminates the multiplexing and polynomial generation logic which is normally required but also sets up appropriate controls for the read and write operations."*

The polynomial generation is required for producing the read error word and the write error words. Clearly, Lemone is directed to a physical checksum scheme.

Accordingly, a person skilled in the art would understand from the teachings of Lemone that error detection is achieved through comparison of error words which are generated in response to a logical function applied to the data being written to and read from a storage device. Furthermore, it should be understood that Lemone desires a low cost and simplified circuit configuration for implementing such error detection. Lemone clearly addresses this problem by using a programmable logic array. However, nowhere does Lemone teach or infer that a logical check is used.

The physical checksum scheme described in Loaiza is similar to the ECC scheme described by Lemone, whereby the "checksum constant" is analogous in function to the "error word". Both are generated by some logic circuitry before data is written to the storage device, and after the data is read from the storage device.

In summary, Appellant acknowledges that Loaiza expresses a motivation to reduce overhead, but this overhead is related to a logical check scheme. A logical check scheme is not used by Lemone, therefore a person skilled in the art could not simplify the ECC circuit configuration of Lemone by implementing the Loaiza logical check scheme. This is due to the fact that the ECC scheme of Lemone does not analyse the logical content of the data, which is the fundamental basis for the logical check used by Loaiza. Therefore, there could not be any motivation to combine the reduced overhead logical check scheme with the physical checksum scheme of Lemone.

(ii) Claimed features absent in Lemone and Loaiza

The Examiner acknowledged in the Final Office Action that Lemone does not teach the claimed features whereby

a first check word is stored in a write accumulator;
a second check word is stored in a read accumulator; and
a compare circuit is connected to the write accumulator and the read accumulator for comparing the first check word to the second check word.

These aforementioned features are recited in each of claims 1, 11, 13 and 21. The Examiner alleged that Loaiza disclosed the aforementioned features that were absent in Lemone. In the Response dated September 6, 2006, the Appellant argued that Laoiza does not disclose, teach or infer the use of a write accumulator and a read accumulator for storing first and second check words respectively, and therefore it followed that Loaiza did not disclose or teach a compare circuit connected to the write accumulator and the read accumulator.

In the Advisory Action dated September 27, 2006, the Examiner responded that the claim limitations of (a) a first check word stored in a write accumulator and (b) a second check word stored in a read accumulator, are disclosed by Loaiza:

> *"Loaiza discloses specifically at column 5 storing the data block in one or more disks, which "suggests" as was well known at the time of the invention that the data block may be divided and stored among a number of disks… Those of ordinary skill in the art at the time the invention was made would recognize that dividing the data block amount its different components such as in the instant case with the data and the checksum is well known."*

Appellant understands that the Examiner equates the write accumulator recited in the claims to the data storage unit 112, which consists of disks 114-118. More specifically, the Examiner suggests that the checksum integrated with the data block of Loaiza (as shown in Figure 5) can be divided and stored in separate disks of a RAID (Redundant Array of Independent Disks). Loaiza states at column 5, lines 12-14 that the data storage unit 112 is implemented as a RAID. It is presumed that the Examiner equates one of the separate disks to the claimed write accumulator, which may store the checksum.

Appellant disagrees, and submits that Loaiza does not disclose the claimed write accumulator because the Loaiza storage system is the same as the Lemone storage system. More specifically, as the Examiner has acknowledged that the Lemone ECC system including a single magnetic disk drive does not disclose a write accumulator, the Loaiza storage system could not disclose a write accumulator either. Appellant's reasoning follows.

A RAID is merely a physical configuration of at least two disk drives which store portions of the data block, or duplicates the data block in each disk drive, in order to increase redundancy and/or performance. Regardless of the specific organization of the data in the RAID, it is well known that the at least two disk drives in a RAID is considered to be a single logical disk drive by the system. In otherwords, a data block which may be divided into portions and stored on

separate individual disks of a RAID, will still appear as a unitary block of data to the system or user. A definition of the RAID system is provided on the Internet at http://www.networkdictionary.com/hardware/r.php. The definition is provided below for convenience:

> *"Redundant Arrays of Independent Disks (RAID) is a type of disk drive with two or more drives in combination for increasing data integrity, fault tolerance, throughput or capacity and performance. RAID provides several methods of writing data across/to multiple disks at once. RAID is one of many ways to combine multiple hard drives into one single logical unit. Thus, instead of seeing several different hard drives, the operating system sees only one. RAID is typically used on server computers, and is usually implemented with identically-sized disk drives. With decreases in hard drive prices and wider availability of RAID options built into motherboard chipsets, RAID is also being found and offered as an option in higher-end end user computers, especially computers dedicated to storage-intensive tasks, such as video and audio editing."* (underlining added)

A RAID is nothing more than a physical extension of the way a single hard disk drive stores a block of data. It is very well known that a single hard drive frequently stores data blocks across different sectors of the hard drive. This is called fragmenting, in which a data block is divided into smaller portions that can be written to available sectors at different address locations. Although a data block may be fragmented, the user still sees the data block as a single logical unit. An explanation of disk fragmenting is provided on the Internet at http://en.wikipedia.org/wiki/Disk_fragmentation. An excerpt of the explanation is provided below for convenience:

> *"For example, files in a file system are often broken up into pieces called blocks. When a disk is new, there is space to store the blocks of a file all together in one place. This allows for rapid sequential file reads and writes. However, as files are added, removed, and changed in size, the disk becomes externally fragmented, leaving only small holes in which to place new data. When a new file is written, or when an existing file is extended, the new data blocks will be scattered out across the disk, slowing access due to seek time and rotational delay of the read/write head."*

The Appellant submits that the RAID of Loaiza is no different than the single magnetic disk disclosed in Lemone, which the Examiner had previously acknowledged did not teach a write accumulator. With respect to his error correcting system, Lemone discusses the process for storing a checkword, called an error word, at column 7, line 67 to column 8, lines 1-2:

> *"Thus, the data words incoming from the data processing system and the write error word generated by the error detection circuitry are written onto the disk."*

Similarly, Loaiza at column 5, lines 34-44 describes the generation of the checksum and its storage with the data:

> *"In one embodiment, prior to causing a block of data to be written to data storage unit 112, application 204 initiates a physical checksum calculation process 220 to generate a checksum value for inserting into the data block. In one embodiment, to generate the checksum value, a logical operation, such as an XOR or ADD operation is performed on the data that is contained in the data block. Based on the vector that results from performing the logical operation, and a desired checksum constant, a checksum value is selected and inserted into the data block."*

Column 5, lines 59-62 of Laoiza describes the storage of the data block, which includes the calculated checksum, in the non-volatile memory:

> *"Disk controller 206 then communicates with disk array unit 210 via network 108 to cause the data block to be stored on one or more disks 114-118."*

Accordingly, whether a data block is stored on a single hard drive or a RAID consisting of multiple hard drives, the data block will be segmented (or fragmented) into portions which are stored in non-sequential locations. Therefore, if the segmentation of the data block by Lemone during storage onto a single hard disk drive is not considered analogous to a write accumulator by the Examiner, then the RAID should not be considered analogous to a write accumulator.

Therefore, Appellant submits that Loaiza does not disclose, teach or suggest a write accumulator. For this first reason, the combination of Lemone and Loaiza fails to disclose all the recited features of claims 1, 11, 13 and 21.

Appellant points out that even if the Examiner's reasoning with respect to the RAID of Loaiza is retro-actively applied to the single disk drive of Lemone, both Lemone and Loaiza still do not teach or disclose the claimed read accumulator.

Loaiza fails to teach or infer that the second checksum is to be stored. In particular, Loaiza does not teach that the second checksum can be stored on the RAID, let alone in a component analogous to the claimed read accumulator. Loaiza merely states at column 5, line 63 to column 6, lines 1-9 that the second checksum generated during read out of the data is compared with the first checksum that was stored with the data.

> *"Thereafter, when the data block is again required by application 204, the data block is retrieved from data storage unit 112 and forwarded to disk controller 206. A physical checksum verification process 224 is then performed on the data block to verify that the data has not been corrupted since the physical checksum calculation process 220 was performed. In one embodiment, to execute the physical checksum verification process 224, the logical operation used by physical checksum calculation process 220 is performed on the data block. The results of*

> *the logical operation are then compared with the checksum stored with the data block to determine whether the integrity of the data block has been maintained since the physical checksum calculation process 220 was performed."*

Nowhere does Loaiza teach that the physical checksum generated during data retrieval from the data storage unit 112 is stored anywhere. The only reasonable inference from the cited passage above would be that the second physical checksum is generated and immediately compared to the previously stored first physical checksum.

Therefore, Appellant submits that Loaiza does not suggest or disclose a read accumulator. For this second reason, the combination of Lemone and Loaiza fails to disclose all the recited features of claims 1, 11, 13 and 21.

VIII. CONCLUSION

Appellant has argued that the Examiner's assessment of obviousness of claims 1, 11, 13 and 21 in view of the combination of Lemone and Loaiza, is improper. In particular, Appellant has argued that

(i) there is insufficient motivation to combine the teachings of Lemone and Loaiza, since the desire to reduce logical check overhead in Loaiza is not applicable to the ECC system of Lemone; and

(ii) both Lemone and Loaiza fail to disclose, teach or infer the use of at least one of a write accumulator and a read accumulator.

Therefore, withdrawal of the Examiner's obviousness rejection to claims 1, 11, 13 and 21 under 35 U.S.C. 103(a) is requested.

Appellant respectfully requests that the Board enter a decision overturning the Examiner's rejection of all pending claims, and holding that the claims are not obvious in view of Lemone and Loaiza.

Respectfully submitted,

JARABEK, Andrew, Louis et al

By: /Gail C. Silver/

Gail C. Silver
Reg. No. 47,945
Borden Ladner Gervais LLP
World Exchange Plaza
100 Queen Street, Suite 1100
Ottawa, ON K1P 1J9
CANADA
Tel: (613) 237-5160
Fax: (613) 787-3558
E-mail: ipinfo@blgcanada.com

SHH/dbm

IX. CLAIMS APPENDIX

1. (Previously Presented) A method including:

generating a first check word based on incoming data;

storing the first check word in a write accumulator;

generating a second check word based on stored data;

storing the second check word in a read accumulator;

comparing the first check word to the second check word by way of a compare circuit connected to the write accumulator and the read accumulator;

generating a comparison result; and

indicating a failure based on the comparison result.

2. (Original) The method of claim 1 wherein generating the second check word occurs at a time subsequent to the data being stored and prior to the data being overwritten.

3. (Original) The method of claim 1 further comprising: generating the second check word during periods of time when the device storing the data is in an idle state.

4. (Original) The method of claim 1 further comprising: synchronizing the generation of the

second check word to the reading and writing of the data.

5. (Original) The method of claim 1 wherein generating the second check word further comprises reading bytes from a selected set of memory locations.

6. (Original) The method of claim 5 wherein the selected set of memory locations includes memory locations included in a single memory.

7. (Original) The method of claim 5 wherein the selected set of memory locations includes memory locations included in multiple memories.

8. (Original) The method of claim 7 further comprising reading the multiple memories simultaneously.

9. (Canceled)

10. (Canceled)

11. (Previously Presented) A method comprising:
generating a first check word based on incoming data to a subset of a plurality of memories;
storing the first check word in a write accumulator;
reading a set of data stored in the subset of the memories;
generating a second check word based on the set of data;
storing the second check word in a read accumulator;
comparing the first check word to the second check word by way of a compare circuit connected to the write accumulator and the read accumulator;
generating a comparison result; and
indicating a failure based on the comparison result.

12. (Original) The method of claim 11 wherein reading a set of data stored in the predetermined subset of the memories includes reading data from multiple memories simultaneously.

13. (Previously Presented) A computer program product tangibly embodied on a computer readable medium, for checking contents of a memory in network switching environment comprising instructions for causing a computer to:

generate a first check word based on incoming data;
store the first check word in a write accumulator;
generate a second check word based on stored data;
store the second check word in a read accumulator;
compare the first check word to the second check word by way of a compare circuit connected to the write accumulator and the read accumulator;
generate a comparison result; and
indicate a failure based on the comparison result.

14. (Original) The computer program product of claim 13 further comprising instructions to: generate the second check word at a time subsequent to the data being stored and prior to the data being overwritten.

15. (Original) The computer program product of claim 13 further comprising instructions to: generate the second check word during periods of time when the device storing the data is in an idle state.

16. (Original) The computer program product of claim 13 further comprising instructions to: synchronize the generation of the second check word to the reading and writing of the data.

17. (Original) The computer program product of claim 13 further comprising instructions to: generate the second check word by reading bytes from a selected set of memory locations.

18. (Previously Presented) The computer program product of claim 13 further comprising instructions to: generate the second check word by reading bytes simultaneously from multiple memory locations.

19. (Canceled)

20. (Canceled)

21. (Previously Presented) A computer program product tangibly embodied on a computer readable medium, for checking contents of a set of memories in network switching environment comprising instructions for causing a computer to:

store data in multiple memories;

generate a first check word based on incoming data to a predetermined subset of the memories;

store the first check word in a write accumulator;

read a set of data stored in the predetermined subset of the memories;

generate a second check word based on the set of data;

store the second check word in a read accumulator;

compare the first check word to the second check word by way of a compare circuit connected to the write accumulator and the read accumulator;

generate a comparison result; and

indicate a failure based on the comparison result.

22. (Original) The computer program product of claim 21 further comprising instructions to: generate the second check word at a time subsequent to the data being stored and prior to the data being overwritten.

23. (Original) The computer program product of claim 21 further comprising instructions to: generate the second check word during periods of time when the device storing the data is in an idle state.

24. (Original) The computer program product of claim 21 further comprising instructions to: synchronize the generation of the second check word to the reading and writing of the data.

25. (Original) The computer program product of claim 21 further comprising instructions to: generate the second check word by reading bytes from a selected set of memory locations.

26. (Previously Presented) The computer program product of claim 21 further comprising instructions to: generate the second check word by reading bytes simultaneously from multiple memory locations.

27. (Canceled)

28. (Canceled)

X. EVIDENCE APPENDIX

None

XI. RELATED PROCEEDINGS APPENDIX

None